import numpy as np

import cv2

import border

import tkinter as tk

from tkinter import filedialog

root=tk.Tk()

root.withdraw()

image\_path=filedialog.askopenfilename() #open dialog-box

image=cv2.imread(image\_path) #open image

image=cv2.resize(image,(1200,900)) #resizing because opencv does not work well with large images(like captured in mobile phones)

orig=image.copy()

gray=cv2.cvtColor(image,cv2.COLOR\_BGR2GRAY) #RGB to Gray-Scale

cv2.imshow("GrayScale",gray)

blurred=cv2.GaussianBlur(gray,(5,5),0) #(5,5) i sthe kernel size and 0 is the sigma that determines the amount of blur

cv2.imshow("Blurred",blurred)

edged=cv2.Canny(blurred,30,50) #MinThreshol=30 & MaxThreshold=50

cv2.imshow("Canny",edged)

contours,hierarchy=cv2.findContours(edged, cv2.RETR\_LIST, cv2.CHAIN\_APPROX\_SIMPLE)

contours=sorted(contours, key=cv2.contourArea, reverse=True)

#following loop enables us to retrieve the boundaries of the required to-be-scanned page

for c in contours:

p=cv2.arcLength(c,True)

approx=cv2.approxPolyDP(c,0.02\*p,True)

if len(approx)==4:

target=approx

break

approx=border.imgborder(target) #to find the endpoints of the to-be-scanned page

pts=np.float32([[0,0],[960,0],[960,720],[0,720]]) #map to 960\*720 target window(4:3 aspect ratio)

top=cv2.getPerspectiveTransform(approx,pts) #get the top or bird-eye view effect of the required to-be-scanned image

dst=cv2.warpPerspective(orig,top,(960,720)) #i have set this according to the test image in landscape mode(change according to the image if portrait)

cv2.imshow("Scanned Output",dst)

cv2.waitKey(0)

cv2.destroyAllWindows()